Journal of Universal Computer Science, vol. 1, no. 1 (1995), 23-34
submitted: 8/10/94, accepted: 19/12/94, appeared: 28/1/95©Springer Pub. Co.

On Implementing EREW Work-Optimally on Mesh of Trees

Ville Leppänen
(University of Turku, Finland
Ville.Leppanen@cs.utu.fi)

Abstract: We show how to implement an $\ell_1 \times n \log n$-processor EREW PRAM work-optimally on a 2-dimensional n-sided mesh of trees, consisting of n processors, n memory modules, and $O(n^2)$ nodes. Similarly, we prove that an $\ell_2 \times n^2 \log n$-processor EREW PRAM can be implemented work-optimally on a 3-dimensional n-sided mesh of trees. By the *work-optimality* of implementations we mean that the expected routing time of PRAM memory requests is $O(1)$ per simulated PRAM processor with high probability. Experiments show that on relatively small ℓ_1 and ℓ_2 the cost per simulated PRAM processor is 1.5–2.5 in the 2-dimensional case, and 2–3 in the 3-dimensional case. If at each step at most $\frac{1}{3}$'th of the PRAM processors make a reference to the shared memory, then the simulation cost is approximately 1. We also compare our work-optimal simulations to those proposed for coated meshes.

Key Words: EREW, mesh of trees, shared memory, simulation, work-optimal, randomized, coated mesh.

Category: C.1.2 C.2.1 F.1.2 F.2.2 G.3.

1 Introduction

PRAM is an abstract model of parallel computation. It consists of p processors and a single shared memory of size m. The shared memory concept of the PRAM is generally believed not to be directly implementable as an extension of the conventional memory technique to the p-port memory technique (this does not seem to hold for relatively small p [Forsell 93]). Therefore, the implementation of PRAM is usually considered on distributed memory machines (DMMs), where processor&memory pairs are connected by some interconnection network [Abolhassan et al. 91, Karp et al. 92, Leppänen and Penttonen 94a, Ranade 91, Valiant 90].

Simulation of PRAM on a 2-dimensional Mesh of Trees (MT) based DMM has been considered previously in [Luccio et al. 88, Pucci 93] probabilistically and in [Luccio et al. 90, Pucci 93] deterministically. The probabilistic simulation of an n-processor EREW PRAM on an n-processor ($O(n^2)$-node) MT is proved to work in time $O(\log n)$ with high probability. The deterministic scheme is respectively proved to work in time $O(\frac{\log^2 n}{\log \log n})$. Thus, the work per simulated PRAM processor is $O(\log n)$ and $O(\frac{\log^2 n}{\log \log n})$, respectively.

In this paper, we show how to decrease the work per simulated processor to $O(1)$ with high probability. We prove this result for both 2-dimensional and 3-dimensional MTs. The method is, of course, increasing the multithreading level of each processor so that the cost caused by routing delay decreases – *i.e.*, we make each of the N real processors to simulate p/N EREW processors, and require that the number of PRAM processors p is sufficiently large. In our simulations, we implement each virtual processor as a light-weight thread (= fixed set of registers). For ease of reference, we call the multithreading level of processors simply by *load*, and increasing the load by *overloading*.

The work-optimality of our simulations can be questioned, since the number of MT-nodes is $O(n^2)$ (or $O(n^3)$ in the 3-dimensional case) while the number of real processors is only $O(n)$ (respectively $O(n^2)$). We adopt the approach taken by Valiant in [Valiant 90] for the work-optimal simulation on the butterfly: If the nodes of the routing machinery are very simple (and fast), then it might be fair to ignore their work-complexity. The nodes of MT are required only to do elementary switching operations, and thus we are willing to ignore their work and hardware complexity. We return to this subject [Section 5].

Next, we give some necessary definitions [Section 2], and describe work-optimal EREW PRAM simulation on 2-dimensional and 3-dimensional mesh of trees [Section 3]. Then, we give experimental EREW simulation results [Section 4], and compare [Section 5] the MT results to those obtained for similar work-optimal simulations on coated meshes [Leppänen and Penttonen 94b]. We conclude [Section 6] by proposing some topics for further research.

2 Definitions

2.1 EREW PRAM

Definition 1. EREW (Exclusive-Read-Exclusive-Write) PRAM model consists of p processors and a shared memory M of size m. Each of the processors $P_0, P_1, \ldots, P_{p-1}$ has some local memory and registers. During one step a PRAM processor can either do a local operation, read a shared memory location, or write to a shared memory location. The phases of each step are executed synchronously, and the next step is not started until all processors have finished the current one. The EREW PRAM does not allow concurrent reading or concurrent writing of a shared memory location. However, a shared memory location may be read and written during the same step. A read operation returns the value of the memory location in question *before* the current step.

2.2 Mesh of Trees

Definition 2. An n-sided d-dimensional *Mesh of Trees* (MT) is a graph, which is based on an n-sided d-dimensional mesh of nodes (without grid edges). For each tower of mesh nodes $\{V_{i_1,\ldots i_{j-1},0,i_{j+1},\ldots,i_d}, \ldots, V_{i_1,\ldots i_{j-1},n-1,i_{j+1},\ldots,i_d}\}$, it contains a complete binary tree $T^j_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_d}$, whose leaves are the nodes of the tower. The edges of complete binary trees are bi-directional, and have a queue of length q packets for both directions. The MT contains no other edges. The degree of MT is $max(3,d)$, and the number of nodes is $|V| = (d+1)n^d - dn^{d-1}$. Respectively, the diameter is $2d \log n$.

In the 2-dimensional case [see Fig. 1], we call T^1_i the i'th *row tree* RT_i, and T^2_i the i'th *column tree* CT_i. In [Luccio et al. 88, Luccio et al. 90], the roots of RT_i and CT_i are joined for each i, but here we do not assume that to be the case. We assume that processor $\mathcal{P}_{i_2,\ldots,i_d}$ is in the root of $T^1_{i_2,\ldots,i_d}$ for each $i_2, \ldots, i_d \in \{0, \ldots, n-1\}$. Similarly, we assume memory module $\mathcal{M}_{i_1,\ldots,i_{d-1}}$ to reside at the root of $T^d_{i_1,\ldots,i_{d-1}}$. Thus, the n-sided d-dimensional mesh of trees consists of $N = n^{d-1}$ processors and memory modules.

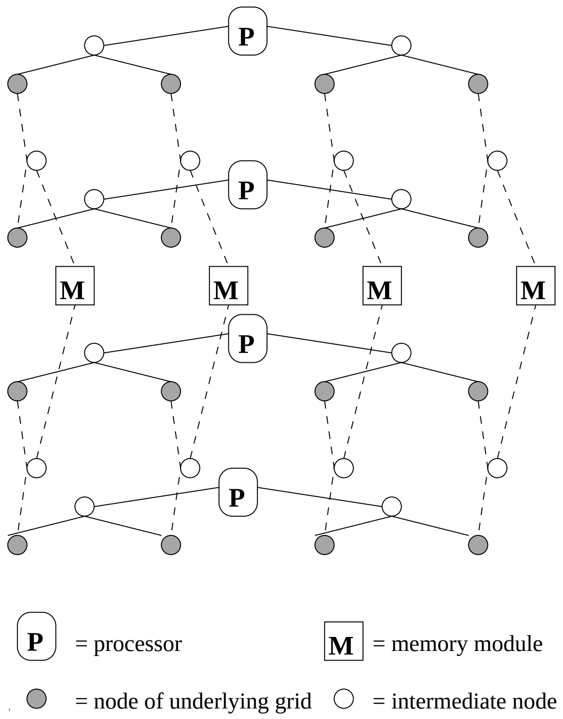

Fig. 1. A 2-dimensional 4-sided mesh of trees.

3 Simulation

Initially, the shared memory is hashed according to some randomly chosen hash function $h \in \mathcal{H}$. Memory references are translated to *read* and *write* packets, which are routed to the memory module on whose custody the referenced shared memory cell is. Each packet is routed along the obvious route as in [Luccio et al. 88, Pucci 93]. The memory modules in turn reply to each read request as they arrive, and route the replies back to the requesting processor. Proper information about the target and the origin are carried in the packets. Before a write packet is "executed", the old value is copied to a backup table (a hash table within each memory module). Those values are used to generate replies to read packets arriving after a write packet with the same target.

For hashing, we use the following family $\mathcal{H}_\zeta$ of polynomial hash functions.

$$\mathcal{H}_\zeta = \left\{ h \middle| h(x) = (\sum_{0 \le i < \zeta} a_i x^i \ mod\ q)\ mod\ m; 0 < a_i < q, m \le q = O(m) \right\},$$

where q is a prime and $\zeta \geq 2$. The family $\mathcal{H}_\zeta$ is not the best possible, because we would like to define mappings $owner : \mathbb{Z}_m \mapsto \{0, \ldots, n-1\}$ and $location : \mathbb{Z}_m \mapsto \mathbb{Z}_{m'}$ by $owner(x) = (h(x)\ div\ m')\ mod\ n$, and $location(x) = h(x)\ mod\ m'$. This does not work in practice, since a randomly chosen $h \in \mathcal{H}_\zeta$ is not bijective. However, the secondary hashing techniques within memory modules (as in [Ranade 91]) can be used to solve the problem. Notice that the serial evaluation time of $h(x)$ is $O(\zeta)$, but if processors have a certain pipeline of length $O(\zeta)$, then the amortized evaluation time can be pushed down to $O(1)$.

Lemma 3. *[Kruskal et al. 90, Corollary 4.20] If a randomly chosen $h \in \mathcal{H}_\zeta$ is used for hashing a set S of unique memory locations into n modules, for which $|S| = s \geq \zeta n/2$, then for all j $(0 \leq j < n)$:*

$$Pr(b_j > s/n + \epsilon) \leq \frac{\zeta}{2}\left(\frac{e\zeta s}{2n\epsilon^2}\right)^{\frac{\zeta}{2}},$$

where $b_j = |\{x \in S \mid h(x) = j\}|$, and $\ln e = 1$.

3.1 2-dimensional Mesh of Trees

In the 2-dimensional case, the routing is straightforward, since the processors are on the roots of row trees and the memory modules are on the roots of column trees. In fact, the path from $\mathcal{P}_i$ to $\mathcal{M}_j$ via mesh node (i, j) is unique. Moreover, if q is sufficiently large, no collisions can happen, when read and write packets traverse "down" along row trees, or when replies traverse down along column trees. Collision, and thus queuing, happens only, when replies traverse "up" along row trees, or read and write packets traverse up along column trees. If s packets are destined to some memory module $\mathcal{M}_j$, then a packet destined to $\mathcal{M}_j$ is delayed (queued) at most $s-1$ times. Lemma 3 gives a good bound for the number of packets destined to each memory module, and consequently we have Theorem 4.

Theorem 4. *For properly chosen (small) constants k and ℓ, there exists such a constant $\alpha \geq 1$ that a 2-dimensional n-processor MT with $\mathcal{H}_{O(\log n)}$ addressing and $q = O(\ell \log n)$ can simulate an $\ell \times n \log n$-processor EREW PRAM work-optimally in $O(\ell \log n)$ expected routing steps with probability at least $1 - n^{-\alpha}$, if $\zeta = k \log n$ and $\ell \log n \geq \zeta/2$.*

Proof. We assume that each processor simulates $\ell \log n$ EREW processors. For the time being, assume that $q = \infty$. According to Lemma 3 ($s = \ell \times n \log n$; $\epsilon = 2s/n$)

$$\begin{aligned} &Pr(b_j > 3s/n) \\ &\leq \frac{\zeta}{2}\left(\frac{e\zeta s}{2n(2s/n)^2}\right)^{\frac{\zeta}{2}} \\ &= \frac{\zeta}{2}\left(\frac{e\zeta}{8\ell \log n}\right)^{\frac{\zeta}{2}} \end{aligned}$$

$$\le \frac{\zeta}{2}\left(\frac{e}{4}\right)^{\frac{\zeta}{2}}$$
$$\le n^{-\alpha'}$$

for some positive constant α', if $\zeta = \Omega(\log n)$. Since b_j tells how many requests memory module $\mathcal{M}_j$ receives at most with high probability, we know that every request reaches its destination in $T_1 = 2\log n + 3\ell \log n$ steps with probability at least $1 - n^{-\alpha'+1}$. Routing the packets back is easier, since each processor receives at most $\ell \log n$ replies. Thus, the last reply is received at most $2\log n + \ell \log n$ steps after some memory module received the last read request.

The queues do not need to be infinite. If $q > 3\ell \log n$, then according to the above reasoning none of the queues becomes full with high probability. Thus, setting $q = 3\ell \log n + 1$ guarantees that the queue length will not affect the routing time with high probability.

How do we know, when to start simulating the next EREW step? We could assume that we first check whether all processors have received all replies. However, we do not actually need such a global control, if we proceed in the following way. Assume that after the last memory reference packet, each processor sends an End-Of-Stream packet. The row tree nodes can spread this EOS-packet to both branches, and respectively the column tree nodes let all other packets go before they combine two EOS-packets, and forward the result upwards. Now, each memory module knows when it has issued the last reply, and can thus send an End-of-Replies packet. Assume that the EOR-packets are transferred in the same way as the EOS-packets. Clearly, each processor can start simulating the next step, when it has received an EOR-packet. In principle, a processor could start simulating the next round right after it has injected its EOS-packet. However, in practice this can cause problems with the coordination of virtual processors. Acting like this, the simulation of at most two consecutive EREW steps are overlapped, but never mixed.

As in [Luccio et al. 88, Pucci 93], we can protect ourselves against some repeatedly occuring bad memory reference patterns, by requiring that the whole shared memory is rehashed, if some memory module receives more than $c \times \ell \log n$ packets for a carefully chosen constant c. Clearly, a memory of size $m = n^\beta$ can certainly be redistributed in time $O(n^\beta + \log n)$. By now we know that if $c \ge 3$, the redistribution takes place at most with probability $n^{-\alpha''}$. Thus, if $\alpha'' > \beta$, the effect on the expected number of routing steps is negligible. $\square$

3.2 3-dimensional Mesh of Trees

To extend the result of Theorem 4 to the 3-dimensional mesh of trees, we only need to describe how to route the packets, and how to keep the simulation of two consecutive PRAM steps separated. Using Lemma 3, it is again easy to prove that each memory module $\mathcal{M}_{i,j}$ receives at most $3\ell \log n$ packets with high probability.

Let us again call those trees, where the processors and the memory modules are connected, the row and the column trees respectively. Let *depth tree* $DT_{x,y}$ denote tree $T^2_{x,y}$. Now, a packet is sent from processor $\mathcal{P}_{i,j}$ to memory module $\mathcal{M}_{k,l}$ so that the packet goes along $RT_{i,j}$ to mesh node $V_{k,i,j}$, then up along $DT_{k,j}$

and down to mesh node $V_{k,l,j}$, and finally up along $CT_{k,l}$ to $\mathcal{M}_{k,l}$. Similarly, replies go back the same way. Notice that it is not wise to put all the packets to go trough the root of some $DT_{x,y}$.

Can we guarantee that there is no congestion in the depth tree nodes? A read or a write packet entering to a depth tree $DT_{k,j}$ is from one of the n processors $\mathcal{P}_{i,j}$ and is destined to one of the n memory modules $\mathcal{M}_{k,l}$, where $i, l \in \{0, \ldots n-1\}$. By Lemma 3 ($s = \ell \times n \log n$; the number of different banks of n memory modules is $n^2/n = n$; $\epsilon = 2s/n$), we know that at most $c \times \ell \log n$ packets enter to $DT_{k,j}$ with probability at most

$$Pr(b_i > 3s/n) \leq \frac{\zeta}{2}\left(\frac{e\zeta}{8\ell \log n}\right)^{\frac{\zeta}{2}} \leq \frac{\zeta}{2}\left(\frac{e}{4}\right)^{\frac{\zeta}{2}} \leq n^{-\alpha'}$$

for some constant α', if $\zeta = \Omega(\log n)$. Clearly, this also sets a sufficiently large upper bound for q.

As in the 2-dimensional case, we can keep the simulation of consecutive steps separate by sending EOS- and EOR-packets. However, we must require that when they traverse the depth trees, they always go via the root. Based on the above discussion, we have Theorem 5.

Theorem 5. *For properly chosen (small) constants k and ℓ, there exists such a constant $\alpha \geq 1$ that a 3-dimensional n^2-processor MT with $\mathcal{H}_{O(\log n)}$ addressing and $q = O(\ell \log n)$ can simulate an $\ell \times n^2 \log n$-processor EREW PRAM work-optimally in $O(\ell \log n)$ expected routing steps with probability at least $1 - n^{-\alpha}$, if $\zeta = k \log n$ and $\ell \log n \geq \zeta/2$.*

3.3 Practical Remarks

It is straightforward to extend the EREW simulation result to higher dimensional mesh of trees. However, finding an efficient layout for d-dimensional ($d > 3$) mesh of trees is obviously very difficult, if not impossible.

We did not pay much attention to the impracticality of family $\mathcal{H}_\zeta$, since we are mainly interested about the routing cost. We believe that simpler families (*e.g.*, $\mathcal{H}_1$) can be used in practice [Engelmann and Keller 93], since the number of all possible different reference patterns

$$\binom{m}{N}$$

is so huge that it is not necessary to guarantee success with high probability for all of them. After all, what is wanted is that in the long run the average simulation time of one PRAM step is $O(\ell \log n)$.

Although the rehashing method proposed earlier in this paper is sufficiently good for asymptotic complexity results, it is likely to be too "rough" in practice. Undoubtly, it is good to have a rehashing mechanism, but its triggering criteria should be chosen very carefully. We believe that one should make the decision on the basis of a long (bad) simulation sequence.

If we ignore the effect of rehashing on the expected routing cost per PRAM processor, by [Section 3.1] and [Section 3.2], we know that the cost is at most

$$(4+5\ell)/\ell = \frac{\ell \log n + 2\log n + 3\ell \log n + 2 \log n + \ell \log n}{\ell \log n}$$

in the 2-dimensional case and $(6+11\ell)/\ell$ in the 3-dimensional case. In practice, we suspect that the cost caused by queuing is not as big as indicated by our naive analysis. Especially, the cost $(2+6\ell)/\ell$ that comes from the depth trees in the 3-dimensional case is too big. In the next section, we confirm this to be the case.

4 Experimental Results

Full details of our routing experiments on the mesh of trees are documented in [Leppänen 94b]. Here, we only give an overview of the test setting and the results.

The integration of processors and the memory modules to the mesh of trees based routing machinery is as described before. We assume that the processors and the routing machinery nodes can send and receive at most one packet in one time unit. We assume that the memory modules can generate a reply in one time unit, and there is a FIFO queue of a fixed length l_q associated to each directed edge. Our experiments indicated that the size of l_q will not significantly affect on the routing time as long as $l_q \geq 2$ [Leppänen 94b]. In the following, $l_q = 2$.

We did not use $\mathcal{H}_\zeta$ to define the destinations of packets, since we did not know how to produce typical access patterns (it makes no sense to apply $\mathcal{H}_\zeta$ to randomly produced access patterns). Instead, we used destinations generated by Unix random function `random`. The packets we perceived as read packets, and thus all the packets were each time routed to their destination and back to their source.

We made about 30 experiments with each chosen parameter combination. Altogether about 2400 routing experiments were conducted on 2-dimensional 64-, 256-, and 512-processor MTs [see Fig. 2], and on 3-dimensional 256-, 1024-, and 4096-processor MTs [see Fig. 3]. We measured only the time to complete a single experiment – as mentioned earlier, overlapping of consecutive steps is likely to decrease the total simulation cost. In each case, the variation of routing times was small. The curves describing our experiments show the dependency of simulation cost c (average simulation time per *Load*) as a function of ℓ ($Load = \ell \times \log n$).

We see that for 2D MT sizes 64, 256 and 512, value $\ell \approx 3$ yields cost $c \approx 5$. When $\ell \approx 8$, then the cost $c \leq 2$. Furthermore, it seems that *the larger the mesh of trees the lower the simulation cost* per processor. Even though our experiments deal only with relatively small MTs, we would like to claim that the simulation cost is very small on large MTs with load $2 \times \log n$.

In the 3-dimensional case, we found out that value $\ell \approx 4$ yields cost $c \approx 4$. When $\ell \approx 7$, then the cost $c \approx 3$. As in the 2-dimensional case, it seems that the larger the mesh of trees the lower the simulation cost.

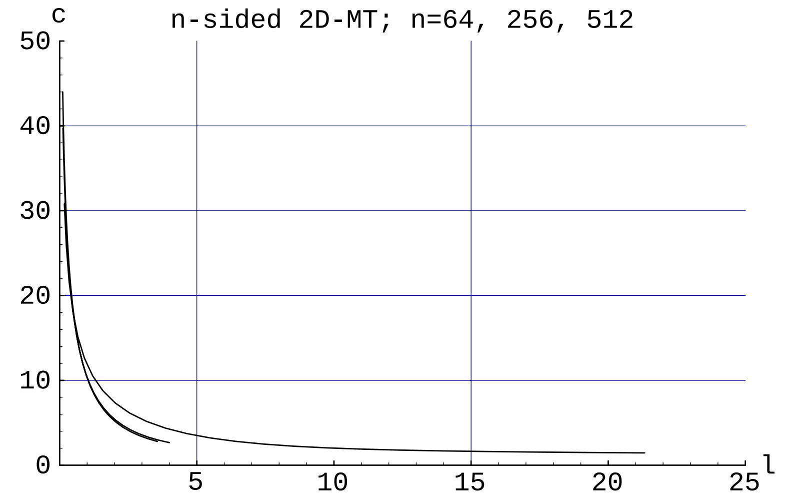

Fig. 2. The simulation cost as a function of ℓ in 2D MT.

The highest and at the same time the longest of the curves represents a 2D MT of size 64. The next highest (and longest) curve corresponds to a 256-processor 2D MT, and the lowest curve represents a 512-processor 2D MT. The Y-axis shows simulation cost c per simulated processor (in terms of routing steps per simulated processor), and the X-axis shows the load as a function of ℓ, where $\ell = Load \div \log n$.

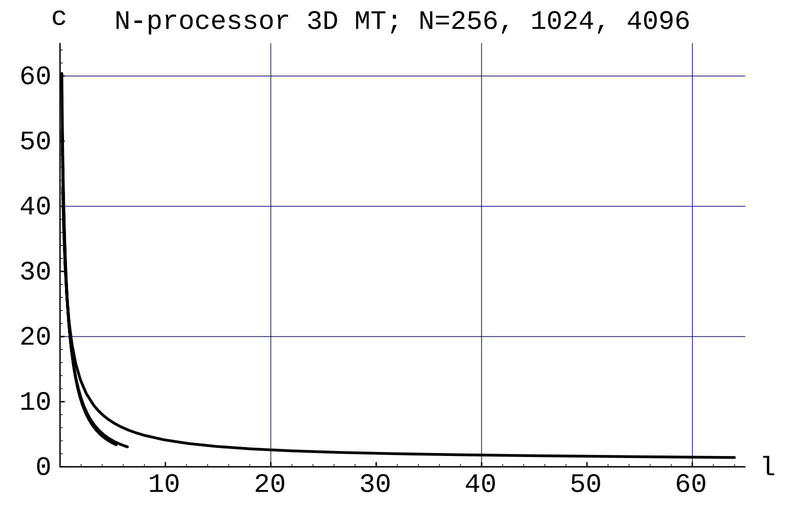

Fig. 3. The simulation cost as a function of ℓ in 3D MT.

The highest and at the same time the longest of the curves represents 3D MT of size 256 processors. The next highest (and longest) curve corresponds to a 1024-processor 3D MT, and the lowest curve represents a 4096-processor 3D MT.

5 Comparison with Coated Mesh

As observed, the simulation cost is very small for the 2-dimensional and 3-dimensional mesh of trees. However, other parameters of EREW PRAM implementations are also important. In the following, we present a comparison with the simulation cost on the coated meshes [Leppänen and Penttonen 94b].

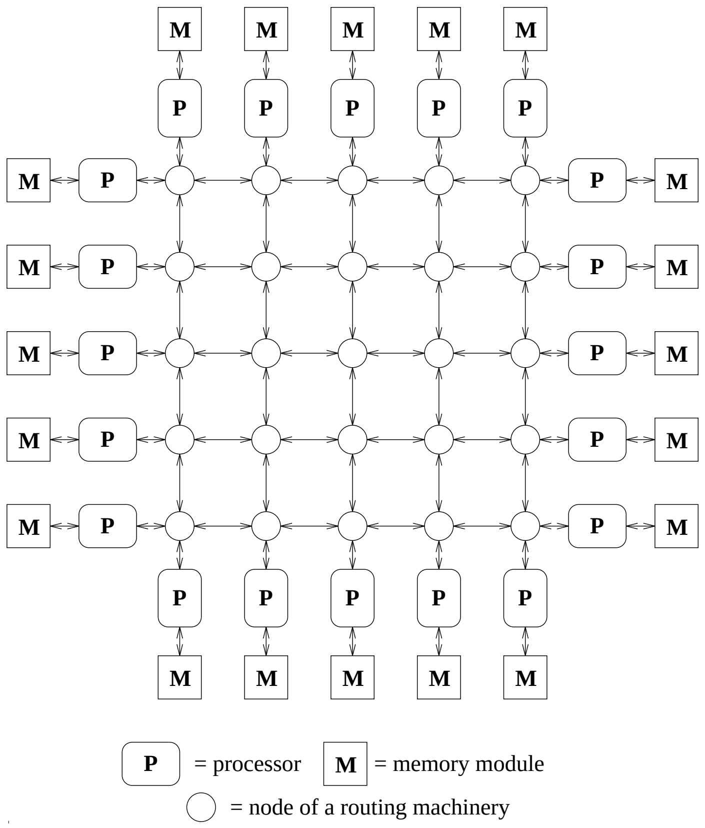

Fig. 4. A 2-dimensional coated mesh with 20 processors.

A coated mesh [see Fig. 4] consists of a mesh connected routing machinery coated with processor&memory pairs. Both the coated mesh and the mesh of trees have a routing machinery of size $O(N^2)$ in the 2-dimensional case, and $O(N^{3/2})$ in the 3-dimensional case. For parameters of our comparison [Tab. 1], we take the routing machinery size with respect to the number of processors and memory modules; simulation cost on a quite moderate load; simulation cost on a heavy load; and the minimum physical distance between logical neighbors. We note that there exist "tricks" to improve efficiency, like integration of the routing machinery nodes; faster clockrate in the routing machinery than in the processors; and delayed memory access operations. All of them can obviously be used to further improve the simulations both cases. We feel that the distance between neighboring nodes is important, since it might limit the clockrate of the routing machinery. So far, increasing the clockrate has been a major source of performance improvements. For the coated mesh structure, we use the experimental results documented in [Leppänen 93, Leppänen 94a].

Property	*MT*	CM	$N = 10^3$	$N = 10^6$
2-dimensional case				
Cost	2–2.5	9–10	*4*	*4*
Load	5–7log N	$\frac{6-7}{16}N$	*6*	*3100*
Cost	1.5	6	*4*	*4*
Load	$20 \log N$	$\frac{30}{16}N$	*9*	*4700*
Number of nodes	$3N^2 - 4N$	$N^2/16$	48	48
Distance	$\Omega(\lceil \frac{2\sqrt{3N^2-2N-2}}{4 \log N} \rceil)$	1	87	43000
3-dimensional case				
Cost	3–4	13–14	*4*	*4*
Load	$3 \log N$	$0.7\sqrt{N}$	1.4	*12*
Cost	1.5	8	*5*	*5*
Load	$60 \log N$	$3\sqrt{N}$	6	*2.5*
Number of nodes	$4N^{3/2} - 5N$	$(N/6)^{3/2}$	56	59
Distance	$\Omega(\lceil \frac{3\sqrt[3]{4N^{3/2}-3N-3}}{3 \log N} \rceil)$	1	5	80

Table 1. Mesh of Trees versus Coated Mesh.

N is the number of real processors in each case. *Distance* tells the lower bound for the minimum (physical) distance between two logical neighbors (measured in routing machinery nodes). To our knowledge, no layout achieving the lower bound is known. *Cost* tells the simulation cost on a given *Load*. The two rightmost columns compare the two PRAM implementations with N processors. An *emphasized* number x means that MT is x times better than CM in this respect. Respectively, plain x means that CM is x times better.

In [Tab. 1], we have chosen two load values for both comparisons. In all cases, the simulation cost depends on the available load in a very similar way. The load values of MT and CM are chosen from similar positions of the load-cost dependency curves [Leppänen 94b, Leppänen and Penttonen 94b]. Especially, we attempted to choose the measure points so that the relative position on the MT curve and on the corresponding CM curve is the same. The first values are chosen from an area, where the load-cost curve begins to show asymptotic behavior, and the second values are chosen from an area were the behavior is asymptotic.

The mesh of trees is clearly better [Tab. 1] in terms of the simulation cost and the load in the 2-dimensional case. In the 3-dimensional case, the mesh of trees is only slightly better in this respect. Moreover, the routing machinery nodes are a little bit simpler in the mesh of trees (less inputs and outputs). However, what is gained in the simulation cost and in the required load, is lost in the size of the routing machinery and in the distance between routing machinery nodes. Especially, in the 3-dimensional case it seems that the coated mesh is actually better than the mesh of trees.

A 10^6-processor 3-dimensional coated mesh has only $\sqrt{N}/6^{1.5} \approx 70$ times more routing machinery nodes than processors. For a corresponding mesh of trees this ratio is about 4000. Remember that this PRAM simulation approach relies on the assumption that the routing machinery nodes are considerably simpler than the processors (and the memory modules). We do not know the actual difference of the routing machinery nodes and the processor&memory pairs in the hardware complexity, but ratio 70 does not seem to be totally unacceptable. Especially, if a bunch of routing machinery nodes (*e.g.*, $8 \times 8 \times 8$) are integrated together to form a building block of a routing machinery.

6 Conclusions and Future Work

We have presented a work-optimal EREW PRAM implementation for the 2-dimensional and 3-dimensional mesh of trees. The simulation uses a novel technique to keep the simulation of consecutive PRAM steps separated. Although the proved simulation costs are small, our experiments show the real simulation costs to be about 2–3 times smaller in practice. We compared the properties of the presented simulations to those proposed for the 2-dimensional and 3-dimensional coated meshes. Neither a mesh of trees nor a coated mesh is strictly better than the other, but our conclusion is that in the 3-dimensional case the coated mesh is better, when all the mentioned properties are considered.

We would like to learn more about the hardware complexity of the routing machinery nodes, and the ability to fast support a large number of virtual processors (how large systolic register set arrays can be built). It would also be interesting to compare these EREW PRAM simulations to those proposed for other logarithmic networks. Extending our work-optimal EREW simulation to an efficient work-optimal CRCW simulation is also an open problem.

References

[Abolhassan et al. 91] Abolhassan, F., Keller, J., Paul, W.J.: "On the Cost-Effectiveness of PRAMs"; Proc. 3rd IEEE Symposium on Parallel and Distributed Computing, ACM Special Interest Group on Computer Architecture, and IEEE Computer Society (1991), 2 – 9.

[Engelmann and Keller 93] Engelmann, C., Keller, J.: "Simulation-Based Comparison of Hash Functions for Emulated Shared Memory"; Proc. PARLE'93 Parallel Architectures and Languages Europe, Springer, LNCS 694 (1993), 1 – 11.

[Forsell 93] Forsell, M.J.: "Are Multiport Memories Physically Feasible?"; Technical Report A-1993-1, University of Joensuu, Department of Computer Science (1993).

[Karp et al. 92] Karp, R.M., Luby, M., Meyer auf der Heide, F.: "Efficient PRAM Simulation on a Distributed Memory Machine"; Proc. 24th Annual ACM Symposium on Theory of Computing (1992), 318 – 326.

[Kruskal et al. 90] Kruskal, C.P., Rudolph, L., Snir, M.: "A Complexity Theory of Efficient Parallel Algorithms"; Theoretical Computer Science, 71 (1990), 95–132.

[Leppänen 93] Leppänen, V.: "PRAM Computation on Mesh Structures"; Technical Report R-93-9, University of Turku, Computer Science Department (1993). Ph.Lic. thesis.

[Leppänen 94a] Leppänen, V.: "Performance of Four Work-Optimal PRAM Simulation Algorithms on Coated Meshes"; Manuscript (1994), submitted for publication.

[Leppänen 94b] Leppänen, V.: "Experimental Results on Simulating EREW PRAM Work-Optimally on Mesh of Trees"; Technical Report R-94-10, University of Turku, Computer Science Department (1994), also appeared as electronic version, anonymous FTP cs.utu.fi, in pub/techreports/1994/R-94-10.ps.Z.

[Leppänen and Penttonen 94a] Leppänen, V., Penttonen, M.: "Simulation of PRAM Models on Meshes"; Proc. PARLE'94 Parallel Architectures and Languages Europe, LNCS 817 (1994), 146 – 158.

[Leppänen and Penttonen 94b] Leppänen, V., Penttonen, M.: "Work-Optimal Simulation of PRAM Models on Meshes"; Technical Report R-94-1, University of Turku, Computer Science Department (1994), submitted for publication.

[Luccio et al. 88] Luccio, F., Pietracaprina, A., Pucci, G.: "A Probabilistic Simulation of PRAMs on a Bounded Degree Networks"; Information Processing Letters, 28 (1988), 141–147.

[Luccio et al. 90] Luccio, F., Pietracaprina, A., Pucci, G.: "A New Scheme for the Deterministic Simulation of PRAMs in VLSI"; Algorithmica, 5, 4 (1990), 529 – 544.

[Pucci 93] Pucci, G.: "Parallel Computational Models and Data Structures"; Technical Report TD-13/93, PhD thesis, Dipartimento di Informatica, Universitá di Pisa – Genova – Udine, Italia (1993).

[Ranade 91] Ranade, A.G.: "How to Emulate Shared Memory"; Journal of Computer and System Sciences, 42 (1991), 307–326.

[Valiant 90] Valiant, L.G.: "General Purpose Parallel Architectures"; Algorithms and Complexity, Handbook of Theoretical Computer Science A (1990), 934–971.

Acknowledgements

The author would like to thank Martti Penttonen for guidance and helpful comments. This work was possible due to a grant provided by the computer science department of the University of Turku.